Server and database resources

In client/server systems complex services are composed of other services. In the following example *Server1* provides the services *buy* and *report*. *Buy* is composed of two other services called in the order. *Order* is called first and its results are waited for, *sales* is called then and it results are waited for. *Report* is implemented locally and it does not need any other services.

```
Server: Server1
    Service: buy
        Calls: order
        Calls: sales
    Service: report
```

While the server is executing a service, it is reserved and cannot accept other service requests. If the required service is not immediately available, the caller waits until is becomes available. Deadlocks may occur if servers wait for each other. In the following example, if someone calls *B_service* and *D_service* simultaneously, a deadlock results. Both servers are busy because they have a service under processing. *Server1* waits for the release of *Server2*, and *Server2* waits for the release of *Server1*. Note that even though the deadlock halts the servers 1 and 2, *Server3* can still continue without failure. Thus, deadlock in this context does not mean that the entire system halts. It means that some servers can never complete the service they are processing.

The system is started by one or more service requests from an outside client. Exported service means that the service can be called from outside the subsystem presented here. Other services are only for internal use and they cannot be requested from outside.

```
Server: Server1
    Service: A_service
    Exported service: B_service
        Calls: C_service
Server: Server2
    Service: C_service
    Exported service: D_service
        Calls: A_service
Server: Server3
    Exported service: E_service
```

(a) Model the following client/server system with a Pr/T net. The only purpose of the model is the detection of deadlocks. What does a deadlock look like in the reachability graph?

```
Server: Server1
    Exported service: order
        Calls: log_order
    Exported service: sales
        Calls: log_sales
    Service: special
        Calls: log_order
        Calls: log_sales

Server: Server2
    Service: retrieve_stock
        Calls: check_delivery
    Exported service: correction

Server: Server3
    Exported service: check_delivery
```

```
        Calls: print
    Service: log_order
        Calls: retrieve_sales

Server: Server4
    Exported service: report
        Calls: correction
        Calls: print
    Service: log_sales
        Calls: check_delivery
    Service: retrieve_sales

Server: Server5
    Exported service: print

Server: Server6
    Exported service: big_order
        Calls: sales
        Calls: special
    Exported service: daily_tasks
        Calls: report
        Calls: print
        Calls: correction
        Calls: print
    Exported service: control
        Calls: retrieve_stock
        Calls: print
```

(b) Present your net model in the PROD description language. Find with PROD at least one execution path leading to a deadlock, or prove that the system cannot deadlock. Hint: Is it necessary to model the *print*-service?

Database locking may also cause the system to deadlock. In the following example, a deadlock results if *Server1* locks *table* concurrently with *Server2* locking *file*. Both servers wait for the other to release its lock.

```
Server: Server1
    Exported service: A_service
        Locks: table
        Locks: file
        Releases: file
        Releases: table
Server: Server2
    Exported service: B_service
        Locks: file
        Locks: table
        Releases: table
        Releases: file
```

(c) The system of exercise (b) is extended with the following database locks. Find deadlocks. How do you explain your observations? Hint: Look at the strongly connected components of the reachability graph, or use PROBE's query language.

```
    Exported service: control
        Locks: database
        Calls: retrieve_stock
        Calls: print
        Releases: database

    Exported service: order
        Locks: database
        Calls: log_order
        Releases: database
```

(d) Are there any other ways than reachability analysis to analyze the above problems?

N.B.: The home assignment is an individual task. No signs of cooperation shall be visible in the returned assignments.

The deadline for the Home Assignment #4 is Nov 24, 2000; use the post box between rooms B336 and B337 in the Computer Science building. You can also return the home assignment by email in PostScript or LaTex formats. The homework assistant is Jukka Honkola (email: Jukka.Honkola@hut.fi).